A Scheme for Collecting Anonymous Data

Shinsuke Tamura, Shuji Taniguchi
Graduate School of Engineering
University of Fukui
Fukui, Japan
tamura@u-fukui.ac.jp

***Abstract*—This paper proposes a scheme for calculating aggregate values of data owned by anonymous entities. Linear mix-nets, an unknown unique number generator, and anonymous tag based credentials efficiently conceal not only identities of data owners but also linkages between data owned by same entities. Then, the scheme enables quick introduction of advancing information technologies to industrial applications while ensuring confidentiality of sensitive data, e.g. a machine maintenance company can monitor states of machines remotely without knowing secrets of machine owners, companies can outsource even their sensitive tasks without worrying about leaks of their secrets. The scheme is also applicable to tasks in e-governance systems such as tax-collections where privacies of individuals must be preserved.**

***Index Terms*— remote maintenance, e-governance, cloud computing, linear mix-net, anonymous tag, anonymous credential.**

I. Introduction

Let us assume a situation where a remote maintenance company calculates the sum of data about machines at factories of a manufacturing company. Where, the sum is calculated for some reasons, but confidentiality of the manufacturing company must be ensured. Therefore, the remote maintenance company must calculate the sum without knowing owners of individual data. Because a set of data owned by a same company include many clues to identify the company, it must calculate the sum without knowing linkages between data owned by same companies either. Same situations exist also in cloud computing and e-governance systems, e.g. a company that carries out its tasks by using cloud computing resources may want to conceal its sensitive data even from mangers of the cloud computing system, and citizens in e-governance systems may not want governments to know their total properties or to link their individual properties.

The anonymous data collection scheme proposed in this paper exploits linear mix-nets, an unknown unique number generator, and anonymous tag based credentials to cope with the above situations securely and efficiently. Namely, one of the linear mix-nets is used to encrypt individual data owned by anonymous entity *P* so that no one except *P* can know correspondences between individual data and *P*, nevertheless by using the unknown unique number generator an authority (e.g. a remote maintenance company) can collect encrypted data of *P*, and the other linear mix-net enables the authority to calculate the sum of data owned by *P* from the collected encrypted forms so that *P* can take required actions, e.g. replace old machines with new ones or pay fees for services provided by the maintenance company. Finally, anonymous tag based credentials enable the authority to identify entities that do not complete the required actions.

Where, although various mechanisms for handling anonymous information have been developed already, they are not efficient enough for data collection systems. For example, it is possible to calculate sums of data without knowing their individual values or their owners when widely used mix-nets [2][3] and encryption schemes such as RSA or ElGamal are combined. Namely, mix-nets conceal linkages between encrypted and decrypted forms of data (this means the owner of data can conceal the correspondence between itself and its data), also RSA and ElGamal enable the authority to calculate the sum of data from their encrypted forms (this means the authority can calculate the sum without knowing individual values). However, mix-nets, RSA and ElGamal are designed to handle data represented as integers, and they are too inefficient to process real numbers that appear in many important applications. Different from usual mix-nets, linear mix-nets in the proposed scheme handle real numbers totally in the same way as integers.

About identifications of entities that do not complete required actions (in the remainder, these entities are called dishonest entities), zero knowledge proof (ZKP) based anonymous credentials are widely used and they satisfactory identify dishonest entities while preserving privacies of honest entities. However, ZKPs are not practical, i.e. they require numbers of challenges and responses [4][5]. Credentials based on anonymous tags require only limited number of challenges and responses and reduce computation volumes drastically.

Then, the proposed scheme enables highly efficient calculation of aggregate values while maintaining confidentialities of individual data and linkages between the data and their owners. In the remainder, it is assumed that the authority calculates sums of data. But slight modifications enable also calculation of polynomial functions of data.

II. System Configuration

Fig. 1 shows 6 parts of the proposed system, i.e. they are member registration, distributed data storage, the 1st and the 2nd linear mix-nets, bulletin board (BB), and dishonest entity identification parts. First of all, the member registration part registers entity *P* as a member of the system, i.e. authority *S*

gives anonymous credential $T_P(0)$ to *P* if it is eligible. At the same time *S* defines a unique random number N_P to give it to *P*, where, N_P is encrypted by multiple independent sub-authorities so that no one except *P* can know its value. Then, *P* stores its individual data $D_P(1)$, $D_P(2)$, ---, $D_P(Q)$ in the distributed data storage together with the credential and the encrypted unique random number without disclosing its identity. In detail, for each data item $D_P(q)$, *P* transforms credential $T_P(0)$ to $T_P(q)$ to disable others to link $D_P(q)$ to other data item $D_P(q_*)$ ($q \neq q_*$), proves its eligibility by showing $T_P(q)$, after that stores $D_P(q)$ together with its encrypted form $E^*(k_*, D_P(q))$, where k_* and g_* below represent secret encryption keys of multiple independent sub-authorities as shown later. Also, P attaches credential $T_P(q)$ and unique random number N_P in form $E(g_*, N_P(q), r_{Pq})$. Here r_{Pq} is a random integer that makes the encryption result unique, i.e. although both $E(g_*, N_P, r_{Pq})$ and $E(g_*, N_P, r_{Pj})$ are decrypted to N_P, entities other than *P* cannot know that $E(g_*, N_P, r_{Pq})$ and $E(g_*, N_P, r_{Pj})$ are the encrypted forms of the same number. Then, no one other than *P* can link $D_P(1)$, ---, $D_P(Q)$.

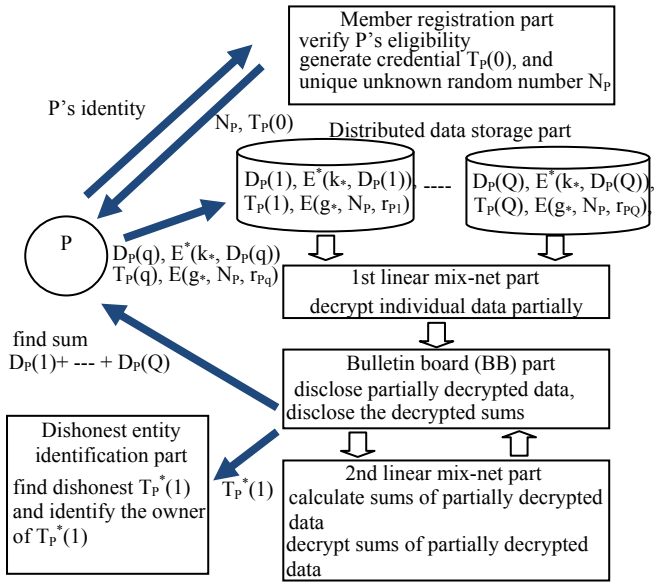

Fig. 1. Configuration of the anonymous data collection system

After that, each triple $\{E^*(k_*, D_P(q)), T_P(q), E(g_*, N_P, r_{Pq})\}$ in the distributed data storage is decrypted (actually $T_P(q)$ is encrypted) by the 1st linear mix-net to be put in the BB while being shuffled with other triples. Here, although $E(g_*, N_P, r_{Pq})$ is decrypted to N_P and $T_P(q)$ is encrypted to $T_P^*(q)$, $E^*(k_*, D_P(q))$ is decrypted only partially into $E^*(k_{Z*}, D_P(q))$, therefore anyone cannot know the correspondence between $E^*(k_{Z*}, D_P(q))$ and $D_P(q)$. But because all data owned by *P* are accompanied by N_P, S can collect *P*'s data $E^*(k_{Z*}, D_P(1))$, ---, $E^*(k_{Z*}, D_P(Q))$ to calculate their sum and attach N_P and $T_P^*(1)$ to the sum. In addition, $E^*(k_{Z*}, x)$ is configured as an additive encryption function, therefore the 2nd linear mix-net can decrypt the sum to $D_P(1) + D_P(2) + \text{---} + D_P(M)$ to be disclosed in the BB together with N_P and $T_P^*(1)$.

As a consequence, *P* can identify the sum of its data in the BB based on N_P (P knows N_P) and can begin the required actions about the sum, but entities other than *P* cannot know the correspondence between *P* and the sum, because N_P is known only to P. Nevertheless, credential $T_P^*(1)$ attached to the sum enables the dishonest entity identification part to identify *P* when it does not complete the required actions without knowing any secret of honest members. Namely as discussed later, used seal of credential $T_P^*(1)$ forces *P* to disclose its identity. Then, if *P* is a manufacturing company and each $D_P(q)$ is the fee for maintaining machine M_q owned by *P* for example, remote maintenance company *S* can force *P* to honestly pay fees for services it provided for machines owned by *P*, on the other hand, *P* can conceal its individual machines and their hours of operations from *S*.

III. Unknown Unique Number Generator

To generate unique secret integer N_P authority *S* is constituted by multiple mutually independent sub-authorities S_1, ---, S_Z, S_{Z+1}, ---, S_J, and member *P* and S_{Z+1}, ---, S_J behave as follows [8]. Firstly, each authority S_h generates its secret integer p(h), and informs authority *S* (representative of S_1, ---, S_J) of $C^{p(h)}{}_{\text{mod B}}$. Where B is a publicly known appropriate large integer, therefore, it is practically impossible for anyone other than S_h to know p(h) from $C^{p(h)}{}_{\text{mod B}}$, i.e. calculating p(h) from $C^{p(h)}{}_{\text{mod B}}$ is a discrete logarithm problem. After receiving $C^{p(h)}{}_{\text{mod B}}$ from each S_h, *S* calculates $C^{NP} = C^{p(Z+1)}C^{p(Z+2)}\text{---}C^{p(J)} = C^{p(Z+1)+p(Z+2)+ \text{---} +p(J)}$ (in the followings notation $_{\text{mod B}}$ is omitted when confusions can be avoided) and asks each S_h to disclose p(h) to *P* when C^{NP} did not appear before. Then, *P* calculates $N_P = p(Z+1)+p(Z+2)+ \text{---} +p(J)$ and finally encrypts it to $E(g_*, N_P, r_{P1})$, $E(g_*, N_P, r_{P2})$, ---, $E(g_*, N_P, r_{PQ})$ by using public encryption keys g_{Z+1}, ---, g_J of S_{Z+1}, ---, S_J. Where, $E(g_*, N_P, r_{Pq}) = E(g_J, E(g_{J-1}, \text{---} E(g_{Z+1}, N_P, r_{Pq}) \text{---}))$, and r_{Pq} is *P*'s secret integer that makes encryption function $E(g_*, x)$ probabilistic as mentioned before.

In the above, each S_h does not know p(j) generated by other authority S_j, therefore no single entity except *P* can know N_P. Because *P* defines r_{P1}, r_{P2}, ---, r_{PQ} independently, no one except *P* can know the linkage between $E(g_*, N_P, r_{Ph})$ and $E(g_*, N_P, r_{Pj})$ when $h \neq j$ either. Also, apparently uniqueness of N_P is ensured, and at the same time confidentiality of N_P is maintained even when same value C^{NP} appears repeatedly, because each S_h knows only p(h).

IV. Linear Mix-net

A linear mix-net is a mix-net, in which linear equation based encryption functions are exploited. Namely as shown in Fig. 2, authorities S_J, ---, S_{Z+1}, S_Z, ---, S_1 in the 1st and the 2nd linear mix-nets decrypt $E^*(k_*, D_P(q)) = E^*(k_J, E^*(k_{J-1}, \text{---} E^*(k_1, D_P(q)) \text{---}))$, member *P*'s data $D_P(q)$ repeatedly encrypted by secret keys k_1, k_2, ---, k_J of S_1, S_2, ---, S_J, into $D_P(q)$ while shuffling individual decryption results with those owned by other members (actually S_Z, ---, S_1 in the 2nd linear mix-net do not need to shuffle their decryption results), therefore no single entity other than *P* can identify the correspondence between $E^*(k_*, D_P(q))$ and $E^*(k_{h*}, D_P(q)) = E^*(k_h, E^*(k_{h-1}, \text{---} E^*(k_1, D_P(q)) \text{---}))$ for any h (i.e. $D_P(q)$ and $E^*(k_{h*}, D_P(q))$.

Here, one of distinctive features of each encryption function $E^*(k_h, x)$ is it can handle real numbers as same as integers, therefore, member *P* and authorities can encrypt and decrypt their relevant data efficiently even the data are real numbers. As another important feature, $E^*(k_h, x)$ is additive, i.e. relation $aE^*(k_h, x)+bE^*(k_h, y) = E^*(k_h, ax+by)$ holds for any real numbers *a* and *b*. Therefore, when the 1st linear mix-net S_J, ---, S_{Z+1} decrypts each $E^*(k_*, D_P(q))$ to $E^*(k_{Z*}, D_P(q))$ and calculates sum $E^*(k_{Z*}, D_P(1)) + \text{---} + E^*(k_{Z*}, D_P(Q))$, the sum coincides with $E^*(k_{Z*}, D_P(1) + \text{---} + D_P(Q))$ as shown at the bottom of Fig. 2, and $\text{Total}_P = D_P(1) + \text{---} + D_P(Q)$ is revealed when $E^*(k_{Z*}, D_P(1) + \text{---} + D_P(Q))$ is decrypted by the 2nd linear mix-net S_Z, ---, S_1. Then, *S* can know that the sum of data owned by some entity is Total_P, but no one except *P* can know the linkage between $D_P(1)$, --- , $D_P(Q)$ or the correspondence between *P* and Total_P.

About Fig. 2, it must be noted that *P* encrypts N_P by public encryption keys of only authorities S_{Z+1}, ---, S_J. This means all decryption results $E^*(k_{Z*}, D_P(1))$, ---, $E^*(k_{Z*}, D_P(Q))$ corresponding to *P* are accompanied by same integer N_P, therefore, S_{Z+1} can collect $E^*(k_{Z*}, D_P(1))$, --- , $E^*(k_{Z*}, D_P(Q))$ to calculate their sum $E^*(k_{Z*}, D_P(1) + \text{---} + D_P(Q))$.

$E^*(k_h, x)$ with the above features can be constructed as below. Namely, $E^*(k_h, x)$ transforms *x* to vector $\{m_1, m_2, \text{---}, m_G\}$ that is calculated as in Eq. 1, where coefficient matrix $\{k(h)_{st}\}$ constitutes secret key k_h of S_h, and $u_1, u_2, \text{---}, u_G$ are real numbers secrets of S_h [6][8]. Then, entities that do not know matrix $\{k(h)_{st}\}$ cannot calculate *x* from $\{m_1, m_2, \text{---}, m_G\}$, but S_h that knows $\{k(h)_{st}\}$ can solve Eq. 1 to know *x*.

$$\begin{aligned} m_1 &= k(h)_{11}x + k(h)_{12}u_1 + \text{---} + k(h)_{1G}u_{G-1} \\ m_2 &= k(h)_{21}x + k(h)_{22}u_1 + \text{---} + k(h)_{2G}u_{G-1} \\ &\text{-----} \\ m_G &= k(h)_{G1}x + k(h)_{G2}u_1 + \text{---} + k(h)_{GG}u_{G-1} \end{aligned} \quad (1)$$

As widely known, linear equation based encryption functions are weak against plaintext attacks, where an entity illegitimately knows secret keys based on known plain text and their encrypted form pairs (e.g. it is easy to calculate $\{k(h)_{st}\}$ in Eq. 1 when G-mutually independent vectors and their decrypted values are given). But in this scheme, S_h itself that encrypts *x* decrypts $\{m_1, m_2, \text{---}, m_G\}$, therefore plain text that includes secret numbers $u_1, u_2, \text{---}, u_{G-1}$ is never disclosed to others. Also, encrypted form of *x* can be constructed as $\{m_1, m_2, d_1, m_3, d_2, \text{---}\}$ while being merged with secret dummy elements $\{d_1, \text{---}, d_{G*}\}$, and m_j can be represented as the sum of multiple elements $\{m_{j1}, m_{j2}, \text{---}, m_{jj*}\}$ if necessary, therefore plain text attacks become extremely difficult.

However different from usual mix-nets, encryption key k_h of $E^*(k_h, x)$ is a secret of S_h, therefore *P* must ask S_1, ---, S_J to encrypt each $D_P(q)$, and as a result, although *P* is anonymous S_h can know the linkage between $D_P(q)$ and partially decrypted form $E^*(k_{h*}, D_P(q))$, because S_h itself calculates $E^*(k_{h*}, D_P(q))$ from $E^*(k_{(h-1)*}, D_P(q))$ at a time when *P* asks authorities to encrypt $D_P(q)$ to $E^*(k_*, D_P(q))$. To disable S_h to know this linkage, *P* constructs $E^*(k_*, D_P(q))$ as follows.

As shown in the upper part of Fig.2, firstly *P* asks S_1, ---, S_J to repeatedly encrypt $D_P(q)$ to $E(k_*, D_P(q)) = E(k_J, E(k_{J-1}, \text{---} E(k_1, D_P(q)) \text{---}))$, where $E(k_h, D_P(q))$ is calculated as linear combinations of $D_P(q)$ and secret random numbers as same as $E^*(k_h, D_P(q))$ shown in Eq. 1. After that *P* generates its secret numbers $a_{1q}(P)$, $a_{2q}(P)$, --- , $a_{Lq}(P)$ and calculates $E^*(k_*, D_P(q))$ as $E^*(k_*, D_P(q)) = E(k_*, D_P(q)) + a_{1q}(P)E(k_*, 0_1) + \text{---} + a_{Lq}(P)E(k_*, 0_L)$. Here, each $E(k_*, 0_t)$ is an encrypted form of 0 that is calculated by S_1, ---, S_J in advance. Then, $E^*(k_*, D_P(q))$ is still decrypted to $D_P(q)$, but S_h cannot know the correspondence between $E(k_{h*}, D_P(q))$ and $E^*(k_{h*}, D_P(q))$, because $a_{1q}(P)$, --- , $a_{Lq}(P)$ are secrets of *P*.

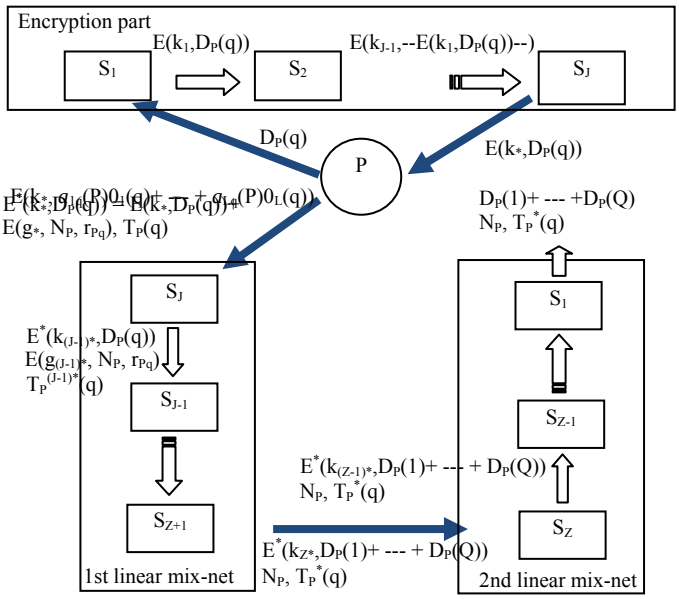

Fig. 2. Linear min-net

There are 2 problems in implementing the above scheme, the one is that S_h may calculate $E(k_*, D_P(q))$ dishonestly, and the other is that *P* may construct $E^*(k_*, D_P(q))$ dishonestly, e.g. *P* must pay more than it is responsible as machine maintenance fees in the former case, and in the latter case *S* cannot collect exact fees from *P*. Fortunately $E(k_*, x)$ and $E^*(k_*, x)$ can be made verifiable because they are additive.

About the former threat conceptually, *S* prepares test bit strings *test*(1), *test*(2), ---, *test*(L_*), and S_1, ---, S_J encrypt them repeatedly to $E(k_*, test(1))$, $E(k_*, test(2))$, ---, $E(k_*, test(L_*))$ in advance. After that at a time when S_1, ---, S_J calculate $E(k_*, D_P(q))$, *P* asks them to decrypt $E(k_*, test) = E(k_*, D_P(q)) + v_{1q}(P)E(k_*, test(1)) + \text{---} + v_{L*q}(P)E(k_*, test(L_*))$ while generating its secret numbers $v_{1q}(P)$, $v_{2q}(P)$ ---, $v_{L*q}(P)$. Then, *P* can convince itself that S_1, ---, S_J are honest if the decryption result coincides with $test = D_P(q) + v_{1q}(P)test(1) + \text{---} + v_{L*q}(P)test(L_*)$. Namely, if $E(k_*, D_P(q))$ is correct, S_1, ---, S_J that know their secret keys can easily decrypt $E(k_*, test)$ to *test*, but if $E(k_*, D_P(q))$ is incorrect, they cannot calculate *test* because they do not know $v_{1q}(P)$, ---, $v_{L*q}(P)$. Actually, $E(k_*, D_P(q))$ is a G-dimensional vector, and each S_h can obtain G-equations to calculate $v_{1q}(P)$, ---, $v_{L*q}(P)$. Therefore, L_* must be greater than G, and this means *P* can obtain many plain

texts and their encrypted forms pairs if L_* is large. Although the scheme is still strong enough, to make the scheme more secure L_* must remain as small as possible. A simple extension makes L_* small enough [8].

The latter threat also can be removed, i.e. *S* can detect dishonest construction of $E^*(k_*, D_P(q))$ and can identify the corresponding dishonest entity *P* in the following way. Firstly, *S* calculates $\mathrm{Sum}(q) = u_{1q}D_1(q) + u_{2q}D_2(q) + \text{---} + u_{\underline{L}q}D_{\underline{L}}(q)$ ($\underline{L}$ is the number of members) and $E^*(k_*, \underline{\mathrm{Sum}}(q)) = u_{1q}E^*(k_*, D_1(q)) + \text{---} + u_{\underline{L}q}E^*(k_*, D_{\underline{L}}(q))$ while generating secret numbers u_{1q}, ---, $u_{\underline{L}q}$, and S_J, ---, S_1 decrypt $E^*(k_*, \underline{\mathrm{Sum}}(q))$. Then, *S* convinces itself that each $E^*(k_*, D_P(q))$ is correct when $E^*(k_*, \underline{\mathrm{Sum}}(q))$ is decrypted to Sum(q). Namely, if $E^*(k_*, D_P(q))$ is an encrypted form of X, $E^*(k_*, \underline{\mathrm{Sum}}(q))$ is decrypted to $\mathrm{Sum}(q)+u_P(X - D_P(q))$. On the other hand, each S_h can know only a partially decrypted form $E^*(k_{h^*}, \underline{\mathrm{Sum}}(q))$, it cannot know $E^*(k_{h^*}, D_P(q))$, i.e. S_h cannot identify the correspondence between $E^*(k_*, D_P(q))$ and $E^*(k_{h^*}, D_P(q))$. Because u_{1q}, ---, $u_{\underline{L}q}$ are secret of *S*, even conspiring entities P_1 and P_2 cannot encrypt their data dishonestly either, i.e. although P_1 can consistently calculate $E^*(k_*, X)$ instead of $E^*(k_*, D_{P1}(q))$ as the encrypted form of $D_P(q)$ if P_2 calculates $E^*(k_*, D_{P2}(q))$-$E^*(k_*, (u_{P1}/u_{P2})(X - D_{P1}(q)))$ instead of $E^*(k_*, D_{P2}(q))$, either P_1 or P_2 that does not know $u_{(P1)q}$ or $u_{(P2)q}$ cannot calculate $(u_{(P1)q}/u_{(P2)q})E^*(k_*, X\text{-}D_{P1}(q)))$.

Once dishonesties are detected, for each triple $\{D_P(q), E(k_*, D_P(q)), T_P(q)\}$, S_J, ---, S_1 decrypt $E(k_*, D_P(q))$, and if $E(k_*, D_P(q))$ is not decrypted to $D_P(q)$, identify *P* as dishonest entity while exploiting anonymous credential $T_P(q)$ as will be discussed later. However after dishonest entities are identified, honest members must ask S_1, ---, S_J to encrypt their data again to conceal linkages between their owning data.

V. ANONYMOUS TAG BASED CREDENTIALS

A. Anonymous Tags

An anonymous tag is a tag and associate tag parts pair $\{T, T^R\}$, i.e. tag owner *P* places T in the tag part and transforms it to $T^R{}_{\mathrm{mod\ B}}$ by its secret integer R to put the result in the associate tag part [7], where as shown in Sec. III, B is an appropriate integer common to all entities and large enough. Also, *P* uses tag $\{T, T^R\}$ while transforming it to $\{T^W{}_{\mathrm{mod\ B}}, T^{RW}{}_{\mathrm{mod\ B}}\}$ by its secret integer W, and entity S_h transforms tags it receives by its secret integer k_h. Therefore, $\{T, T^R\}$ changes its form as $\{T^{W(k1)}, T^{RW(k1)}\}$, $\{T^{W(k1)(k2)}, T^{RW(k1)(k2)}\}$ and $\{T^{W(k1)(k2)(k3)}, T^{RW(k1)(k2)(k3)}\}$ in this order when it is transformed by *P* and 3 entities S_1, S_2, S_3. Then, anonymous tags satisfy the following requirements, i.e.

1) anyone except *P* cannot identify *P* from its tag,
2) anyone except *P* cannot know that different forms of tag $\{T, T^R\}$ are owned by a same entity, unless all relevant entities conspire with each other, and
3) *P* can identify its tag without knowing secrets of others.

About requirement 3), *P* can identify its tag by transforming the tag part by its secret R, e.g. *P* can identify $\{T^{W(k1)(k2)}, T^{RW(k1)(k2)}\}$ as its tag by calculating the associate tag part value from tag part value $T^{W(k1)(k2)}$ as $(T^{W(k1)(k2)})^R$. Here, it is apparent that *P* does not need to know secret integers k_1, k_2, --- to identify its tags.

B. Anonymous Credentials Based on Anonymous Tags

Let T_P, *k* and *c* be integers defined by authority *S*, and R and *w* be secret integers defined by member *P*. Then, provided that d_1 and d_2 are 2 secret signing keys of *S* and $S(d_1\|d_2, x)$ is RSA signature pair $\{S(d_1, x) = x^{d1}{}_{\mathrm{mod\ B}}, S(d_2, x) = x^{d2}{}_{\mathrm{mod\ B}}\}$, signature pair $S(d_1\|d_2, T_P{}^{R+1}K_wC_w{}^R{}_{\mathrm{mod\ B}})$ is an anonymous tag based anonymous credential generated by *S* and given to *P* (it must be noted that RSA signing function is multiplicative, i.e. relation $S(d, x)S(d, y) = S(d, xy)$ holds). Here, T_P, *k* and *c* are publicly known integers, and different from *k* and *c* that are common to all credentials T_P and R are unique to credential $S(d_1\|d_2, T_P{}^{R+1}K_wC_w{}^R)$. Regarding K_w and C_w, *P* calculates them as $K_w = k^w{}_{\mathrm{mod\ B}}$ and $C_w = c^w{}_{\mathrm{mod\ B}}$, respectively based on *k* and *c*. Also uniqueness of *P*'s secret integer R can be maintained in the same way as in Sec. III.

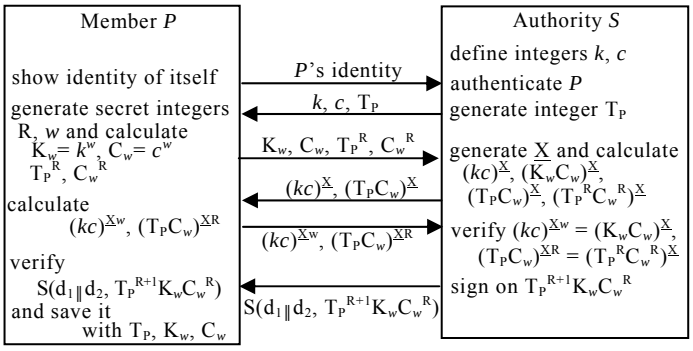

Fig. 3. Issuing anonymous credentials

Fig. 3 shows the procedure in which *S* issues $S(d_1\|d_2, T_P{}^{R+1}K_wC_w{}^R)$ to *P*. Firstly, after verifying authenticity of *P* based on *P*'s identity, *S* defines integer T_P, and *P* calculates $K_w = k^w$, $C_w = c^w$ and pair $T_P{}^R$ and $C_w{}^R$ while using its secrets *w* and R, where $\{T_P, T_P{}^R\}$ constitutes an anonymous tag. Then *S* generates credential $S(d_1\|d_2, T_P{}^{R+1}K_wC_w{}^R)$ to give it to *P*. Here, it must be noted that although $T_P{}^R$, k^w, c^w and $C_w{}^R$ are disclosed, to know *w* and R is practically impossible for *S* as mentioned before. Nevertheless, *S* can confirm that *P* had calculated K_w and C_w as *k* and *c* to the power of same *w* through the scheme of Diffie and Hellman [1], i.e. *S* generates secret integer $\underline{X}$ to calculate $(kc)^{\underline{X}}$ and $(K_wC_w)^{\underline{X}}$, and asks *P* to calculate $(kc)^{\underline{X}w}$ while showing $(kc)^{\underline{X}}$. If *P* does not know *w* that satisfies relation $K_w = k^w$ and $C_w = c^w$, it cannot calculate $(kc)^{\underline{X}w} = (K_wC_w)^{\underline{X}}$ from $(kc)^{\underline{X}}$ without knowing *S*'s secret $\underline{X}$ (actually, it is easy to find α and β that satisfy $\alpha\beta = (kc)^w$ by defining α arbitrarily and calculating $\beta = (kc)^w/\alpha$, and *P* can report α and β to *S* instead of k^w and c^w while calculating $(\alpha\beta)^{\underline{X}}$ that is equal to $(kc)^{\underline{X}w}$. However in this case, *P* that does not know γ or δ that satisfies $\alpha = k^\gamma$ or $\beta = c^\delta$ cannot calculate $\alpha^{\underline{Y}} = k^{\gamma\underline{Y}}$ or $\beta^{\underline{Y}} = c^{\delta\underline{Y}}$ when *S* generates secret integer $\underline{Y}$ and asks *P* to calculate them from $k^{\underline{Y}}$ or $c^{\underline{Y}}$). In the same way, *S* can confirm that $T_P{}^R$ and $C_w{}^R$ are equal to T_P and C_w to the power of same unknown integer R.

After having obtained $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$, by generating secret integer W, calculating $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^W = S(d_1 \| d_2, (T_P^{R+1} K_w C_w^R)^W)$ and showing it together with T_P^W, K_w^W, C_w^W and C_w^{RW}, *P* can convince any entity *V* that it is an eligible entity ensured by *S* without disclosing its identity as shown in Fig. 4. Namely, *V* verifies that $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^W$ is the legitimate signature pair on $(T_P^{R+1} K_w C_w^R)^W$ by using public verification key pair d_{1*} and d_{2*} (pair $S(d_1 \| d_2, x)$ is consistent when not only $S(d_1, x)$ and $S(d_2, x)$ are consistent signatures but also they are decrypted into same *x*), decomposes it into T_P^W, T_P^{RW}, K_w^W and C_w^{RW} based on the information given by *P*, and confirms that *P* had calculated $K_w^W = k^{wW}$ and $C_w^W = c^{wW}$ based on integers *k*, *c* and same unknown secret integer *w*W in the same way as *S* had verified pair $\{K_w, C_w\}$ in Fig. 3.

Then, because no one other than the legitimate holder of $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$ knows R, *V* can determine *P* is eligible when *P* knows R. Where *P* can prove that it knows R without disclosing R itself also through the scheme of Diffie and Hellman [1], i.e. *V* calculates $(T_P^W C_w^W)^X$ and $(T_P^{RW} C_w^{RW})^X$ while generating its secret integer X, and *P* that receives $(T_P^W C_w^W)^X$ calculates $D = ((T_P^W C_w^W)^X)^R$ by using its secret R, then finally, *V* determines that *P* knows R when relation $D = (T_P^{RW} C_w^{RW})^X$ holds. Namely, although it is easy to calculate $(T_P^{RW} C_w^{RW})^X$ from $(T_P^W C_w^W)^X$ for an entity that knows R, for entities that do not know R calculating $(T_P^{RW} C_w^{RW})^X$ is practically impossible.

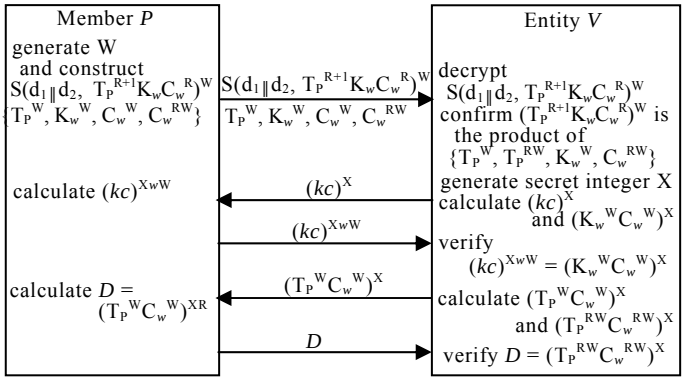

Fig. 4. Verifying anonymous credentials

Now, the above procedures are characterized by the following 4 properties, i.e.

1) *P* can obtain credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$ from *S* only through legitimate ways and only when it is eligible,
2) only *P* that knows integer R can prove the ownership of $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$,
3) no one except *P* can identify *P* from its credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^W$. Provided that W_1, ---, W_N are different integers secrets of *P*, no one other than *P* can know that $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{W1}$, ---, $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{WN}$ are different forms of same credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$ either, and
4) *P* that shows credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^W$ can prove its ownership only when it calculates $D = (T_P^W C_w^W)^{XR}$ as $(T_P^W C_w^W)^X$ to the power of exactly R.

In addition, any entity can verify the validity of $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$ because verification keys d_{1*} and d_{2*} are publicly known and $(T_P^{R+1} K_w C_w^R)^W$ is decomposed into T_P^{RW}, T_P^W, K_w^W and C_w^{RW} by *P* itself. Here, property 4) in the above are ensured because *V* examines whether K_w^w and C_w^w are calculated as publicly known integers *k* and *c* to the power of same unknown integer *w*W, and T_P^{RW} and C_w^{RW} are equal to T_P^W and C_w^W to the power of same unknown integer R, in the same way as *S* in Fig. 3 examines relation $K_w = k^w$ and $C_w = c^w$. Also a signature pair disables entities to forge a credential while generating α, β and δ arbitrarily and calculating $\alpha^\beta k^\delta c^{\beta\delta}$ to transform it by a publicly known verification key.

About anonymity of credentials, when 2 entities *P* and P_* use $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^W$ and $S(d_1 \| d_2, T_{P*}^{R*+1} K_{w*} C_{w*}^{R*})^{W*}$ while defining integer pairs $\{w, W\}$ and $\{w_*, W_*\}$ in the way relation $wW = w_* W_*$ holds by chance, *S* (it is assumed that authority *S* itself verifies credentials) can detect this fact by duplicated appearances of K_w^W $(= k^{wW})$ and if *S* is conspiring with P_*, it can know *w*W $(= w_* W_*)$ by asking it to P_*. But *S* cannot identify *P* from *w*W, i.e. *S* that knows only *w*W cannot extract W and calculate T_j^W for each T_j it had assigned to member P_j to compare the result with T_P^W that *P* shows together with $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^W$.

C. Anonymous Tag Based Credentials in the Proposed Scheme

One of advantages of anonymous tag based credentials is member *P* can show credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$ repeatedly while changing its forms without being detected by others that its showing forms were generated from same $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$. Namely, provided that *P* assigns different values to W_1, ---, W_N, no one except *P* can know that $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{W1}$, ---, $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{WN}$ are shown by *P*. While exploiting this property, credentials $T_P(0)$, $T_P(1)$, ---, $T_P(Q)$ in Sec. II can be implemented as $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)$, $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{W1}$, ---, $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{WQ}$, respectively.

As another feature relates to data collection systems, anonymous tag based credentials enable authority *S* to identify dishonest members of course without knowing privacy of honest entities. To identify dishonest members, when *P* puts data $D_P(q)$ in the distributed data storage while showing credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{Wq}$, *S* memorizes pair $\{U_P(q), U_P(q)^R\}$ as a data registration record. Here, *S* defines $U_P(q)$ as an integer unique to data $D_P(q)$. $U_P(q)^R$ is the used seal of tag $\{T_P, T_P^R\}$ and *S* asks *P* to calculate it based on credential $S(d_1 \| d_2, T_P^{R+1} K_w C_w^R)^{Wq}$. Under these settings, if *S* detects that no entity completes actions about the sum accompanied by $\{U_P(1), U_P(1)^R\}$, each member P_* is requested to calculate $U_P(1)^{R*}$ from its credential $S(d_1 \| d_2, T_{P*}^{R*+1} K_{w*} C_{w*}^{R*})$ and $U_P(1)$ in dishonest data registration record $\{U_P(1), U_P(1)^R\}$. Then, *S* can identify P_* as the entity responsible for the actions when $U_P(1)^{R*}$ coincides with $U_P(1)^R$ (i.e. when $R = R^*$). On the other hand, P_* can conceal the linkage between it and its data if it is honest, because it is practically impossible to know $U_{P*}(1)^{R*}$ that P_* had left in the data registration records from $U_P(1)^{R*}$.

In the above, *S* can force *P* to honestly calculate $U_P(q)^R$ at a time when *P* registers $D_P(q)$ as below. Namely, *S* asks *P* to

calculate $D = \{(T_P{}^W C_w{}^W)^X\}^R$, $A = U_P(q)^R$ and $B = \{(U_P(q)T_P{}^W C_w{}^W)^{XY}\}^R$ while showing $(T_P{}^W C_w{}^W)^X$, $U_P(q)$ and $(U_P(q)T_P{}^W C_w{}^W)^{XY}$, where X and Y are *S*'s secret integers. After that, *S* examines whether relations $D = (T_P{}^{RW} C_w{}^{RW})^X$ and $B = (A^{XY})D^Y$ hold or not. Then, if *P* calculates *A* dishonestly as $U_P(q)^Q$ (Q ≠ R), because *D* must be equal to $\{(T_P{}^W C_w{}^W)^X\}^R$ (according to property-4 in the previous subsection, if *P* calculates *D* differently it cannot show the ownership of $\{T_P, T_P{}^R\}$), *P* must calculate *B* so that it coincides with $(A^{XY})D^Y = (U_P(q)^{QXY})(T_P{}^{RW} C_w{}^{RW})^{XY})$, but *P* that does not know X or Y cannot calculate $U_P(q)^{QXY}$. In the same way, *S* can force P_* to honestly calculate $U_P(q)^{R*}$, when *S* detects existences of dishonest members.

But actually data registration record $\{U_P(1), U_P(1)^R\}$ is encrypted by S_J, ---, S_{z+1} together with $T_P(1) = S(d_1 \| d_2, T_P{}^{R+1} K_w C_w{}^R)^{W1}$ as shown in Fig. 2. Namely, the 2nd linear mix-net transforms $\{U_P(1), U_P(1)^R\}$ to $\{U_P(1)^{(eJ)(ej-1)---(ez+1)}{}_{\text{mod B}}, U_P(1)^{R(eJ)(ej-1)---(ez+1)}{}_{\text{mod B}}\} = \{U_P(1)^{e*}{}_{\text{mod B}}, U_P(1)^{Re*}{}_{\text{mod B}}\}$ by secret keys e_J, ---, e_{z+1} of S_J, ---, S_{z+1}. Therefore, dishonest data registration record $\{U_P(1), U_P(1)^R\}$ in the above must be replaced by $\{U_P(1)^{e*}, U_P(1)^{Re*}\}$.

Used seals solve also a problem, in which P_* that conspires with *S* and obtained $D = (T_P{}^W C_w{}^W)^{XR}$ that was calculated by *P* impersonates *P*, the owner of credential $S(d_1 \| d_2, T_P{}^{R+1} K_w C_w{}^R)$. Namely, although P_* can know $S(d_1 \| d_2, T_P{}^{R+1} K_w C_w{}^R)^W$ and $D = (T_P{}^W C_w{}^W)^{XR}$ after *P* had used $S(d_1 \| d_2, T_P{}^{R+1} K_w C_w{}^R)^W$, and *S* and P_* can jointly generate integers X^* and W^* so that P_* can use $S(d, T_P{}^{R+1} K_1 K_2{}^R)^{WW*}$ while calculating $D^{W*X*} = (T_P{}^W C_w{}^W)^{XRW*X*}$ that is consistent with R, either P_* or *S* that does not know R cannot calculate $U_{P*}(m)^R$ (m > n) consistently (here, it is assumed that *P* had left used seals $U_P(1)^R$, $U_P(2)^R$, ---, $U_P(n)^R$ before). Although P_* can leave $U_P(n)^R$ that *P* had calculated before or inconsistent value $\underline{U}$ as its used seal, *S* must reject $U_P(n)^R$ because it is shown repeatedly, also *S* cannot impute the liability of this dishonesty by using $\underline{U}$ because it cannot identify *P* from $\underline{U}$.

By exploiting used seals *S* can also limit the numbers of times that members can use same credentials. Namely, in the credential verification procedure shown in Fig. 4, *P* is requested also to declare n, the number of times that it had used $S(d_1 \| d_2, T_P{}^{R+1} K_w C_w{}^R)$ before, and *S* calculates used seal $U(n)^R$, where $U(n)$ is an integer defined by *S* and common to all members but unique to n. Then, because $U(n)^R$ is unique to pair {*P*, n}, *S* can reject excessively shown $S(d_1 \| d_2, T_P{}^{R+1} K_w C_w{}^R)$, i.e. *S* memorizes pair $\{n, U(n)^R\}$ and rejects *P*'s credential when *P* declares a value more than the limit as n or when pair $\{n, U(n)^R\}$ had appeared already.

As the conclusion of this section, when compared with ZKP based anonymous credentials, which require numbers of interactive or non-interactive challenges and responses between *P* and *S*, procedures in this section require *P* to calculate only values $D = (T_P{}^W C_w{}^W)^{XR}$, $A = U_P(q)^R$ and $B = \{(U_P(q)T_P{}^W C_w{}^W)^{XY}\}^R$. Therefore, anonymous tag based credentials enable developments of highly efficient data collection systems. Also, although all entities are required to carry out the dishonest entity identification procedure when dishonest events are detected, inconveniences caused by *S*'s inquiries are mitigated if the procedures are included in payment processes for services.

VI. Protecting the Scheme from other Dishonesties

As other kind of dishonest behaviors, member *P* may put its data $D_P(q)$ with an encrypted number that cannot be decrypted to N_P. For example, when *P* puts $D_P(q)$ with an encrypted number that is decrypted to N_{P*} or to invalid value N_X, member P_* becomes the one that is responsible for actions corresponding to $D_P(q)$ or authority *S* cannot correctly collect data from *P*. But, *S* can easily detect this dishonesty as duplicated appearances of N_{P*} for $E^*(k_{Z*}, D_P(q))$ and $E^*(k_{Z*}, D_{P*}(q))$ in the BB, or $E^*(k_{Z*}, D_P(q))$ in the BB accompanied by invalid N_X. Also *P* is identified as a liable member by using a credential attached to $E^*(k_{Z*}, D_P(q))$.

VII. Conclusion

A scheme for collecting data owned by anonymous entities and calculating their aggregate values while preserving privacies of individual entities is proposed. Although only summations were considered as aggregation operations, slight modifications of the scheme enable authorities to calculate also general polynomials of anonymous data [8]. In addition, computation volumes can be maintained as low as that for summations when the polynomial functions are not so complicated. Then, it becomes possible to use the scheme in various applications such as medical records processing in addition to those in e-governance and cloud computing systems.

References

[1] W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Trans. On Information Theory, Vol. IT-22, No. 6, pp.644-654, 1976.

[2] D. Chaum, "Untraceable Electronic Mail, Return Address and Digital Pseudonyms," Communications of the ACM, vol. 24, no. 2, pp. 84-88, 1981.

[3] P. Golle and M. Jakobsson, "Reusable Anonymous Return Channels," Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society, WPES '03, ACM, pp. 94-100, 2003.

[4] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya and H. Shacham, "Randomizable Proofs and Delegatable Anonymous Credentials," Proc. of the 29th Annual International Cryptology Conference on Advances in Cryptology, pp. 108-125, August 2009.

[5] S. F. Shahandashti and R. Safavi-Naini, "Threshold Attribute-based Signatures and their Application to Anonymous Credential Systems," Proc. of the 2nd International Conference on Cryptology in Africa: Progress in Cryptology, pp.198-216, 2009.

[6] S. Tamura and T. Yanase, "A Mechanism for Anonymous Credit Card Systems," IEEJ Trans. EIS, Vol. 127, No.1, pp.81-87, 2007.

[7] S. Tamura, H. A. Haddad, K. Kouro, H. tsurugi, K. MD. R. Alam, T. Yanase and S. Taniguchi, "An Information System Platform for Anonymous Product Recycling," Journal of Software, Vol.3 No.3, pp.46-56, 2008.

[8] S. Tamura, "Anonymous Security Systems and Applications: Requirements and Solutions," Information Science Reference, 2012